A more complete understanding of the present invention, as well as further features and advantages of the present invention, will be obtained by reference to the following detailed description and drawings.

Brief Description of the Drawings

FIG. 1 illustrates a cache thrashing reduction system in accordance with the present invention;

FIG. 2 illustrates a thrashed set detector in accordance with a first exemplary embodiment of the present invention;

FIG. 3 is a sample set miss table that contains information about the sets that recently experienced misses; and

FIG. 4 is a schematic block diagram of an exemplary implementation of a concurrent symmetric static-pairs scheme for coupling a thrashed set and an augmentation set.

Detailed Description

FIG. 1 illustrates a cache thrashing reduction system 100 in accordance with the present invention. The present invention improves performance of an N-way associative cache 110 by automatically detecting thrashing in a set, such as sets 2 and 3, and then reducing thrashing in the thrashed set(s) by providing one or more augmentation frames 150 as additional cache space. In this manner, the trashed sets can remain in the cache longer, improving their hit rate. The augmentation frames 150 are obtained, for example, by mapping the blocks that map to a thrashed set to one or more additional, less utilized sets. While the present invention is illustrated with a two-way set associative instruction cache that has two frames at each set address, the present invention may be incorporated into all cache organizations (data or instruction), as would be apparent to a person of ordinary skill in the art. The cache thrashing reduction system 100 and cache 110 can be part of a digital signal processor (DSP), microcontroller, microprocessor, application specific integrated circuit (ASIC) or another integrated circuit.

Detection of Thrashed Sets

Thrashed set detection may be based on the individual miss rate of a set, its miss rate relative to other sets, the addresses of the blocks involved in misses on the set, or a combination of the foregoing. These approaches employ increasingly complex logic and are not all explored here. An exemplary approach is based on an assumption that a set that is experiencing a high miss rate may be experiencing thrashing. Two approaches to thrashing detection are presented that are based on this assumption. The first approach uses more logic than the second approach and is oriented toward caches that are highly associative (four or more frames per set), while the second approach uses less logic and is oriented toward caches that are less associative, or when cost and area are important design criteria.

First Detection Approach

The first approach, discussed further below in conjunction with FIG. 2, associates a miss counter 210 and an access counter 220 with one or more sets. The accesses of a given set are counted and the miss rate is determined by comparing the number of misses experienced during a given number of accesses. However, this implementation may require more logic than is desirable.

A reduction in logic is achieved by counting accesses to a group of sets rather than to individual sets, as shown in FIG. 2. For example, sets may be grouped into pairs and accesses to either set in a pair (counter-pairs) may be counted. Sets may be paired, for example, by their set indexes. For example, a "pair-index" may be derived from a set-index by ignoring its least significant bit and sets with the same pair-index are paired. Sets in these counter-pairs hold contiguous blocks (if set indexes are derived from contiguous address bits). Thus, they are more apt to experience similar access rates than sets paired by other means.

Assume that accesses to counter-pairs are counted by pair-access-counters 220 that are four-bit counters, which wrap to 0x0 after reaching 0xF. The value 0xF is thus achieved by the counter 220 every 16 accesses. When this occurs, the values in the miss-counters 210 of the sets in the counter-pair are examined and then reset to zero. Assume that a miss-counter 210 is a three-bit counter that counts to seven and then saturates. If all bits of the miss-counter 210 are binary ones when examined, the associated set is assumed to be experiencing thrashing. During the last 16 accesses to the set-pair, the set experienced seven misses. Depending on the

distribution of accesses between the two sets in the pair, the set experienced a hit rate between 0% (if it experienced seven accesses) to 44% (if it experienced 16 accesses). If the sets experienced equal numbers of accesses, i.e., eight each, the hit rate of the set is 12%. A low hit rate in this range indicates probable thrashing.

In addition to the miss counters 210 (one per set), and counter-pair access counters 220 (one for two sets), there is a cache access counter 230 to count accesses to the entire cache. This counter 230 produces periodic signals that query thrashed set detection logic and reset counters associated with the sets. The logical function of the first approach for thrashed set detection is illustrated in FIG. 2 for a counter-pair, A and B.

Second Detection Approach

The first approach employs a significant number of counters to detect a thrashed set. In some cache implementations, an approach that uses less logic may be desirable. In the following approach, only the misses and accesses relative to sets that have recently experienced a miss are counted. This significantly decreases the logic associated with thrashed set detection compared to the first approach, which counts the misses for each set in the cache.

Variations of this approach are possible. One exemplary implementation is presented here to illustrate the technique. FIG. 3 is a sample set miss table 300 that contains information about the sets that recently experienced misses, one entry per set. For purposes of illustration, the SMT 300 discussed below has eight entries, one for each of the last eight sets to experience a miss. Each entry in the table 300 is comprised of a set index field 310 that records a cache index of a set, a miss counter 320, an access counter 330, and a valid bit 340. Entries invalidate themselves periodically under certain conditions, making room in the table for new entries.

The operation of the SMT 300 is illustrated using a cache that has 128 sets, with 7-bits in a set index. Also assume that set miss counters are three-bit counters and that set access counters are four-bit counters. The exemplary SMT 300 may be implemented as a content addressable memory (CAM) that is accessed with a set index. When a set is accessed, the SMT 300 is queried to determine if the set's index is in one of the entries in the SMT 300. If the set index is in the SMT 300, and the set experiences a cache hit, the access counter 330 for the associated entry is incremented. If the set is in the SMT 300 and the set experiences a cache miss,